Midterm Examination for *Introductory Lectures on Optimization*

Tianyu Fan
3200100667

February 10, 2023

Question 1. Based on the small datasets *abalone*, *bodyfat*, and *housing*, training a *ridge regression* model with algorithms Gradient Descent, Conjugate Descent, and quasi-Newton method respectively.

Requirement:

1. Implement algorithms (Gradient Descent, Conjugate Descent, and quasi-Newton method) using c/c++ programming language.
2. In this answer sheet, please briefly introduce the ridge regression, the training algorithms that you implement, experimental settings, and experimental results (please illustrate with diagrams the Mean Squared Error of different algorithms at each iteration, and analyze the results in light of what you have learned in this course).
3. Compress the pdf file of this answer sheet and the c/c++ program into a zip file, and submit it.

Remark. datasets can be downloaded from `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html`.

Report of Question 1:

1 Introduction

In this section, we briefly introduce the ridge regression and the training algorithms.

1.1 ridge regression

The ridge regression mainly solves the problem that the ordinary least squares(OLS) method cannot solve when the data have covariance or pathological data are biased.

Taking mean square error(MSE) as loss, out goal is to minimize the $RSS = \sum(\hat{y}_i - y_i)^2$.

Thus, our goal can be written as:

$$J_\omega(\omega) = \arg\min_\omega \sum(y_i - x_i\omega_i - b)^2 = \arg\min_\omega (y - X\omega)^T(y - X\omega)$$

During linear regression using OLS, if write the equation in matrix form and solve it, we will have

$$\hat{\omega} = (X^T X)^{-1} X^T y$$

However, in reality, X^TX is often not a full rank matrix or the linear correlation between some columns is relatively large.

Therefore, in ridge regression, we use

$$\hat{\omega}_{ridge} = (X^TX + \lambda I_n)^{-1}X^Ty$$

During ridge regression, in order to penalize the large $\hat{\omega}$, using

$$J_\omega(\omega) = \arg\min_{\omega} \sum (y_i - x_i\omega_i - b)^2 + \frac{\lambda}{2}\sum ||\hat{\omega}_i||^2$$

Unlike the unbiased estimation of linear regression, ridge regression has the advantage of its biased estimation, which tends to shrink some of the coefficients more toward zero. Therefore, it can alleviate the multiple co-linearity problem, as well as the overfitting problem. However, since the coefficients are not contracted to 0 in ridge regression, but made smaller overall, the explanatory power of the model is greatly reduced in some cases, and it cannot fundamentally solve the multiple co-linearity problem.

1.2 Gradient Descent

Gradient descent is a commonly used algorithm in machine learning, taking an important role as optimization algorithm for solving. The basic idea is to keep approximating the optimal point, and the direction of optimization at each step is the direction of gradient.

During ridge regression, our goal is to minimize the loss: $L = \frac{1}{n}((y - X\omega)^T(y - X\omega)) + \frac{\lambda}{2}\omega^T\omega$.

In order to find the optimal ω, we first calculate the grad of L:

$$\nabla L = -\frac{2}{n}X^T(y - X\omega) + \lambda\omega$$

Therefore, we can follow the step of algorithm 1 to solve the optimal ω.

Algorithm 1 Gradient Descent during ridge regression

Input: *iteration*, lr,ω,X,y,λ
Output: ω

```
k=0;
while k < iteration do
    ∇L = −(2/n) X^T (y − Xω) + λω
    ω = ω − lr ∗ ∇L
end while
return ω
```

1.3 Conjugate Descent

Each update of Gradient Descent(GD) generally changes the values of all dimensions of the ω, so that ω obtained after each iteration is not necessarily the best goal in the previous search direction.

To compensate for this shortage, now we briefly introduce the Conjugate Descent(CD).

For a quadratic function$f = \frac{1}{2}X^TQX - X^Tb$, $Q = Q^T$, the best direction to search is its conjugate direction. That means, CD only searches in the direction that affects one dimension at a time, so the

second search does not change the first dimension of ω, i.e., it does not affect the result of the first search.

What can be proved is that the set of conjugate vectors we need is the set of column vectors of the square root inverse of the Hessian matrix of the target quadratic type.

1.3.1 Using gradient to obtain direction

Then, the key is to obtain the conjugate direction. In the project, I choose to use gradient as a guidance for obtaining the direction.

There are two requirements for constructing search directions, one is that all search directions are conjugate to each other, and the other is that search direction d_k is just a linear combination of gradient g_k and last direction d_{k-1}.

Let $d_k = -g_k + \beta_{k-1} d_{k-1}$, noticed that $d^T k Q d_k = 0$, then we have $\beta_{k-1} = \frac{g_k^T Q d_{k-1}}{d_{k-1}^T Q d_{k-1}}$.

1.3.2 FR-CG

The above-mentioned analysis is under the assumption that the function needed to be optimized is a quadratic function, which is usually difficult to get.

In 1964, Fletcher and Reeves propose FR-CG, which is used in this project. They replaced the original algorithm with the following equation:

$$\beta_k = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$$

Therefore, we can follow the step of algorithm 2 to solve the optimal ω.

Algorithm 2 FR-CG

Input: *iteration*, ω,X,y,λ
Output: ω

k=0;
while $k < iteration$ **do**
  $g_{now} = -\frac{2}{n} X^T (y - X\omega) + \lambda\omega$
  if $k == 0$, $d = -g_{now}$
  $a = \arg\min_a L(\omega + a * d)$
  $\omega = \omega + a * d$
  $g_{last} = g_{now}$
  $g_{now} = -\frac{2}{n} X^T (y - X\omega) + \lambda\omega$
  $\beta = \frac{g_{now}^T g_{now}}{g_{last}^T g_{last}}$
  $d = -g_{now} + \beta * d$
end while
return ω

1.4 Quasi-Newton

1.4.1 Newton's method

In order to introduce the quasi-newton method, then, we firstly introduce the newton method.

Using taylor expand, we have

$$f(x') = f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)\Delta x^2$$

$$\nabla f = f'(x) + f''(x)\Delta x = 0$$

$$\Delta x = -\frac{f'}{f''}$$

$$x^{(t+1)} = x^t - \frac{f'}{f''} = x^t - \frac{g}{h}$$

1.4.2 BFGS

The Hesse matrix h in Newton's method is computationally intensive to find the inverse when it is dense, and there may be no inverse (when the Hesse matrix h is non-positive definite). The proposed Newton's method replaces the Hesse matrix in Newton's method with a matrix that does not contain second-order derivatives and then does a one-dimensional search along the search direction.

In Newton's method, we have direction $d = -H^{-1}g$, so the key is to find another matrix B^{-1} to approximate H^{-1}.

Using sherman-morrison algorithm, we formulate B^{-1} as follows:

$$B_{t+1}^{-1} = (I_n - \frac{\Delta x_t \Delta g_t^T}{\Delta x_t^T \Delta g_t})B_t^{-1}(I_n - \frac{\Delta g_t \Delta x_t^T}{\Delta x_t^T \Delta g_t}) + \frac{\Delta x_t \Delta x_t^T}{\Delta x_t^T \Delta g_t}$$

Therefore, we can follow the step of algorithm 3 to solve the optimal ω.

Algorithm 3 BFGS

Input: *iteration*, ω,X,y,λ
Output: ω

k=0;B=I;
while $k < iteration$ **do**
  if $k == 0$, $g_{now} = -\frac{2}{n}X^T(y - X\omega) + \lambda\omega$
  $d = -B * g_{now}$
  $a = \arg\min_a L(\omega + a * d)$
  $\omega = \omega + a * d$
  $g_{last} = g_{now}$
  $g_{now} = -\frac{2}{n}X^T(y - X\omega) + \lambda\omega$
  $\Delta g = g_{now} - g_{last}$,$\Delta\omega = a * d$
  $B^{-1} = (I_n - \frac{\Delta x \Delta g^T}{\Delta x^T \Delta g})B^{-1}(I_n - \frac{\Delta g \Delta x^T}{\Delta x^T \Delta g}) + \frac{\Delta x \Delta x^T}{\Delta x^T \Delta g}$
end while
return ω

2 Experiment

2.1 Datasets

In this section, we briefly introduce the datasets used in the project. The details are included in table 1. In the table, "nums" means the number of datas, and "Missing Value" indicates whether there are missing data in the dataset.

In this project, we use 0 to fill the missing value.

Datasets	nums	features	Missing Value
abalone	4177	8	Yes
bodyfat	252	14	No
housing	506	13	No

Table 1: Datasets

2.2 Settings

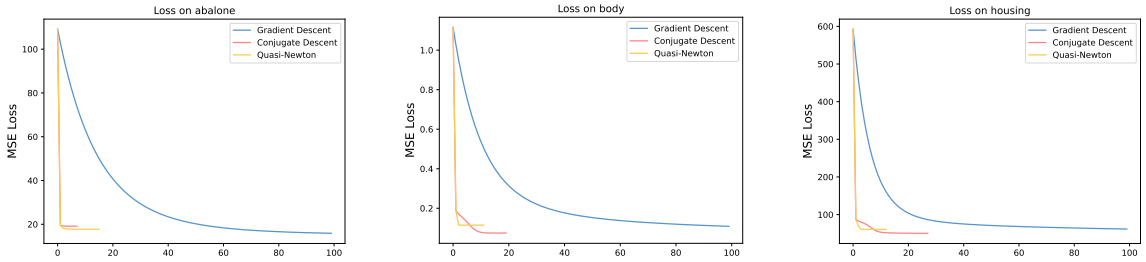
As mentioned above, this project use FR-CG to realize Conjugate Descent, and BFGS to realize Quasi-Newton's method.

During the training process, we set a mechanism of early-stop. To be specific, if the loss does not drop for ten consecutive epochs, exit the process.

2.3 Results and Analysis

2.3.1 Algorithm Comparison

Firstly, we visualize the MSE in fig 1.



(a) abalone (b) bodyfat (c) housing

Figure 1: MSE loss on different datasets

It is obviously to observe that, Conjugate Descent(CD) and Quasi-Newton(QN) need much less epochs to converge than Gradient Descent(GD). And usually(on bodyfat and housing), CD can get a lower loss than the two other methods.

However, less epochs to converge does not mean less time. Table 2 is the average time consumption under different original values on the dataset abalone. We can observe that, though GD usually need much more epochs to converge, it actually the most time saving. We argue that it is mainly because CD and QN need linear search, which is time-expensive.

Methods	Gradient Descent	Conjugate Descent	Quasi-Newton
Time(s)	0.182 ± 0.174	11.161 ± 5.626	14.588 ± 5.138

Table 2: Datasets

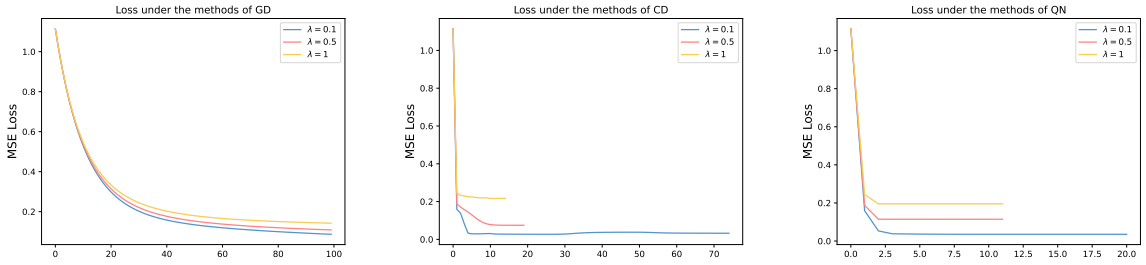
2.3.2 Hyper-parameter Analysis

In this section, we analyze the hyper-parameter sensitivity of different methods. All the following results were trained on bodyfat under different hyper-parameters and methods. We mainly discuss λ and the initial value of ω.

Fig 2 shows the loss with different λ. We can get the following observations:

1.Usually large λ means more epochs to converge.

2.Usually small λ can make a smaller MSE loss.



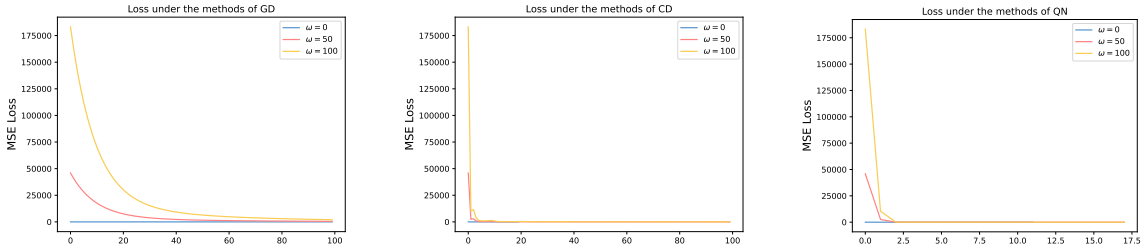
(a) Gradient Descent (b) Conjugate Descent (c) Quasi-Newton

Figure 2: MSE loss under different λ

Fig 3 shows the loss with different initial value of ω, and fig 4 is the final 20% epochs of every results(in order to make a clearer observation). We can get the following observations:

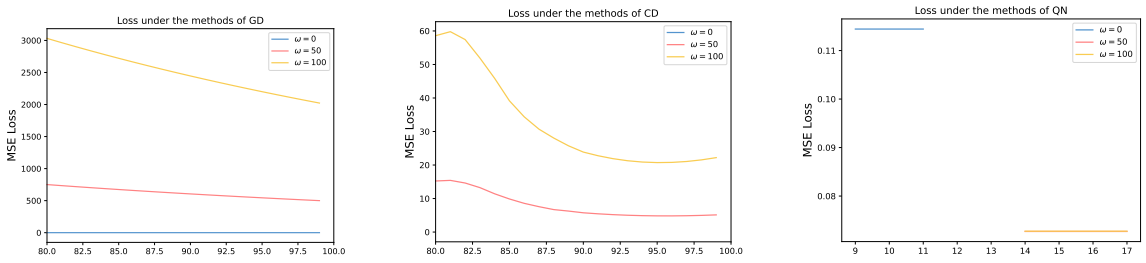
1.The inital value of ω is almost independent of the speed of convergence.

2.Different initial value of ω can lead to different results. However, a rise in initial value of ω does not necessarily lead to a rise or fall in this result, but rather shows a stochastic relationship



(a) Gradient Descent (b) Conjugate Descent (c) Quasi-Newton

Figure 3: MSE loss under different ω_0



(a) Gradient Descent (b) Conjugate Descent (c) Quasi-Newton

Figure 4: MSE loss under different ω_0

3 Conclusion

□